Adapting to Concept Drift in Credit Card Transaction Data Streams Using Contextual Bandits and Decision Trees

Dennis J. N. J. Soemers,[1] Tim Brys,[1] Kurt Driessens,[2] Mark H. M. Winands,[2] Ann Nowé[1]
[1]Vrije Universiteit Brussel, [2]Maastricht University
[1]{dsoemers, timbrys, anowe}@vub.ac.be, [2]{kurt.driessens, m.winands}@maastrichtuniversity.nl

Abstract

Credit card transactions predicted to be fraudulent by automated detection systems are typically handed over to human experts for verification. To limit costs, it is standard practice to select only the most suspicious transactions for investigation. We claim that a trade-off between exploration and exploitation is imperative to enable adaptation to changes in behavior (concept drift). Exploration consists of the selection and investigation of transactions with the purpose of improving predictive models, and exploitation consists of investigating transactions detected to be suspicious. Modeling the detection of fraudulent transactions as rewarding, we use an incremental Regression Tree learner to create clusters of transactions with similar expected rewards. This enables the use of a Contextual Multi-Armed Bandit (CMAB) algorithm to provide the exploration/exploitation trade-off. We introduce a novel variant of a CMAB algorithm that makes use of the structure of this tree, and use Semi-Supervised Learning to grow the tree using unlabeled data. The approach is evaluated on a real dataset and data generated by a simulator that adds concept drift by adapting the behavior of fraudsters to avoid detection. It outperforms frequently used offline models in terms of cumulative rewards, in particular in the presence of concept drift.

1 Introduction

Fraudulent credit card transactions result in significant costs for companies if they are not detected early (Delamaire, Abdou, and Pointon 2009). Credit card transactions can be viewed as a data stream, where large quantities of data arrive over time. It is typically assumed that human experts can investigate and provide accurate labels (indicating whether transactions are fraudulent or genuine) for a small number of transactions (e.g., 100) every day. Additionally, it is assumed that labels automatically become available for all other transactions after a longer delay (Dal Pozzolo et al. 2015a). For example, it can be considered safe to assume that a transaction is valid if the card holder did not report it as fraudulent after seven days. Fraudsters can change their behavior over time in an attempt to avoid detection by a system trained to detect older fraudulent behaviors (Dal Pozzolo et al. 2014a). This problem, where the distribution of a data stream is not stationary, is known as *concept drift*.

Existing Fraud Detection Systems (Dal Pozzolo et al. 2014a; 2015a; Lebichot et al. 2016) typically use an ensemble of Machine Learning models trained offline. Every day at midnight, new models are trained on new sets of labeled data that have become available since the previous night. Large quantities of data become "labeled" with a long delay, and labels of smaller amounts of data from the previous day are provided by human experts. The transactions labeled by human experts in these systems are simply the K instances with the highest predicted reward, where K is the number of transactions that can be investigated per day. The reward of a transaction is often chosen to be equal to the monetary transaction amount if it is indeed fraudulent, or 0 otherwise. This is the primary evaluation criterion used in this paper. The reward is sometimes also chosen to be independent of the monetary transaction amount (Dal Pozzolo et al. 2014a).

In this paper, we aim to address two issues with the systems as described above. The first issue is that experts only provide short-term feedback by investigating transactions that are already predicted to be highly rewarding. Suppose that fraudsters can create concept drift by changing their behavior such that models predict low rewards for their transactions. Those transactions are not selected for investigation, which means that there is no short-term feedback with labels for those instances. Models trained during the following night are unlikely to make better predictions for similar instances, and it may take up to a week before the card holders are expected to have noticed and reported the fraudulent transactions. The second issue is that, even if concept drift is detected early, it still takes until the following night before a new model is trained that uses this new knowledge.

We propose to use an incremental Regression Tree learner (Ikonomovska 2012) to train a model to predict the rewards of transactions. It is computationally cheap to update this model directly when new labels are provided by experts, making it unnecessary to wait until the next night. Additionally, we propose that the problem of selecting which transaction to send to an expert for investigation can be modeled as a Contextual Multi-Armed Bandit (CMAB) problem (Zhou 2015). This model is not feasible when every instance is considered as a separate arm, but becomes feasible by viewing every leaf node of the Regression Tree as a single arm containing multiple instances. The CMAB formalization enables the use of a variety of algorithms that provide a balance

between *exploration* and *exploitation*. In the context of this paper, exploration can be viewed as selecting instances that are expected to be informative when updating the model, and exploitation can be viewed as selecting instances that are expected to have a high reward (i.e. to be fraudulent).

The main contribution of this paper is the idea that a Regression Tree (RT) can be used to create arms in settings where there are too many if every individual instance is viewed as an arm. This enables the use of CMAB algorithms to provide a trade-off between exploration and exploitation in such settings. Additionally, a new variant of a CMAB algorithm is proposed, which takes the structure of the RT into account when predicting rewards. Finally, the RT learner is enhanced with an idea based on Semi-Supervised Learning (Zhu 2005), which enables it to also learn from unlabeled data. The proposed approach is evaluated using a publicly available dataset of real credit card transactions. The ability to adapt to changes in fraudster behavior is furthermore evaluated using a simulator, which was built to generate realistic data, but also enables fraudsters to adapt their behavior.

2 Preliminaries

This section formalizes the problem setting discussed in this paper, and the (Contextual) Multi-Armed Bandit setting.

Let (x_i, y_i) denote a credit card transaction with a unique index i, where $x_i \in \mathbf{R}^d$ is a d-dimensional real-valued feature vector, and $y_i \in \{0, 1\}$ is a binary class label. $y_i = 0$ denotes that transaction i is genuine, and $y_i = 1$ denotes that it is fraudulent. Because the class labels for uninvestigated transactions are not known, we sometimes use only x_i to refer to a transaction. At discrete time steps $t = 1, 2, \ldots$, a human expert becomes available and can investigate a new transaction. At these time steps, the system should select one transaction out of all uninvestigated transactions observed so far, and assign it to the expert. The index of the transaction selected at time t is denoted by i_t. The reward obtained per time step t is defined as $r(t) = amount(x_{i_t}) \times y_{i_t}$, where $amount(x_{i_t})$ is a feature in x_{i_t} that denotes the monetary amount of the transaction. The primary goal is to maximize the cumulative reward $R(\tau) = \sum_{t=1}^{\tau} r(t)$ collected over a large number of time steps τ.

In a Multi-Armed Bandit (MAB) problem, there is a set of K arms, where each arm is associated with an unknown reward distribution. At discrete time steps t, a system selects one of the K arms to pull, which provides a reward sampled from the reward distribution associated with that arm. The goal is to maximize the cumulative reward over a long period of time. In the Contextual MAB (CMAB) setting, there are context vectors available at every time step t which are assumed to be related in some unknown way to the rewards provided by arms in that time step. The problem setting in this paper can be viewed as a CMAB problem where every transaction corresponds to a separate arm. The main problem with this view is that the number of arms grows too rapidly.

3 Related Work

Existing fraud detection systems for credit card transactions are described in (Delamaire, Abdou, and Pointon 2009; Dal Pozzolo et al. 2014a; 2015a; Lebichot et al. 2016). One of the main differences is that those approaches only adapt to concept drift by training new models offline at midnight, whereas the approach described in this paper updates in real-time. Another important difference is that this paper proposes not only to use human experts for exploitation, but also for exploration. An approach with an incremental model that can be updated in real-time was evaluated in (Dal Pozzolo et al. 2014b), but they assumed that the data stream to learn from is completely labeled. Finally, the majority of the existing approaches do not use Semi-Supervised Learning (SSL) techniques to enable learning from the large amounts of unlabeled data. The main exception is (Lebichot et al. 2016), in which unlabeled data is used in a learning step for feature engineering, but not for the training of a predictive model.

Two related problem settings are Active Learning (AL) (Dasgupta 2011) and Active Search (AS) (Garnett et al. 2011). The goal in AL is to select instances to use for training an accurate model in such a way that the number of selected instances is minimized. This task is similar to exploration in the problem setting of this paper, but ignores exploitation. The goal in AS is to select as many instances as possible that, after selection, turn out to belong to one particular class of interest (typically the $y = 1$ class). This is equivalent to the cumulative reward function in the problem setting of this paper if it is assumed that every transaction has exactly the same monetary amount. However, the existing literature for AS focuses on offline datasets or algorithms which are not feasible to run in real-time.

A combination of Decision Trees and CMABs was also recently proposed in (Elmachtoub et al. 2017). They assume that there already is a CMAB problem with well-defined arms, and use Decision Trees to model the relationship between context vectors and sampled rewards. In this paper, a Decision Tree is used to create arms and enable the use of any CMAB algorithm in a setting where there were previously no well-defined arms.

4 Creating CMAB Arms Using Decision Tree

If the problem of selecting which transaction to investigate can be modeled as a CMAB problem, we can use any existing CMAB algorithm to address the trade-off between exploration and exploitation. The main difficulty in applying CMAB algorithms is that there is no natural concept of "arms" in this setting. It is not computationally feasible to treat every transaction as a separate arm in real-time.

The number of arms can be reduced by creating clusters of transactions. Suppose that all transactions are divided among K clusters. The problem can now be viewed as a CMAB problem with K arms, by sampling transactions x_k from every cluster $k \in [1, K]$ in every time step t, and using the feature vector x_k as context vector for arm k at time t.

CMAB problems are typically easy if the distributions of rewards per arm are easily separable, and difficult if different arms have overlapping distributions of rewards. Let $r(x_i) = amount(x_i) \times y_i$ denote the reward that would be obtained if x_i were investigated. Suppose that a Regression Tree $\mathcal{T}$ is trained to predict the reward values $r(x_i)$ for transactions x_i. A learning algorithm for $\mathcal{T}$ typically attempts to

create splits in the tree such that every leaf node is assigned transactions x_i that have similar reward values $r(x_i)$. This is done, for example, by creating splits that maximally reduce the standard deviation among rewards. This means that, if an accurate tree $\mathcal{T}$ can be trained in this way, the resulting CMAB problem with the leaf nodes of $\mathcal{T}$ as arms becomes an "easy" CMAB problem. This insight is the main reason for using a Decision Tree to create clusters of transactions, instead of unsupervised clustering algorithms. In particular, the Regression Tree learner implemented for this paper is based on FIMT-DD (Ikonomovska 2012). This is a Regression Tree learner that can efficiently be updated incrementally in real-time using feedback from human experts.

5 Tree-Boosted Bootstrap Thompson Sampling

A number of different CMAB algorithms (Li et al. 2010; Agrawal and Goyal 2013; Eckles and Kaptein 2014) were evaluated on CMAB problems with leaf nodes as arms in preliminary experiments. In these experiments, Bootstrap Thompson Sampling (BTS) (Eckles and Kaptein 2014) has been found to perform best. This algorithm is used in the remainder of the paper. An extensive evaluation of other CMAB algorithms is outside the scope of this paper.

Bootstrap Thompson Sampling

Pseudocode for a variant of BTS is described in Algorithm 1. It is initialized with a set of J models (for example, $J = 100$ linear regression models), and a cumulative reward $R(\tau) = 0$. At every time step t, context vectors x_a are observed for every arm a. BTS randomly selects one model j_t out of the J models, and uses it to predict rewards for all x_a at time t. It greedily plays the arm with the highest predicted reward, resulting in a reward signal $r(t)$. The context vector x_{a_t} of the played arm, and the reward $r(t)$, can be used to update models. In BTS, they are not used to update each of the J models in every step, but only some of the J models. This can, for example, be done by giving every model a probability of 0.5 to be updated in every time step. By training each of the J models on only a bootstrap of the data, it is expected that different models will start disagreeing with each other over time. This results in exploration, in the sense that some models will greedily play arms that other models may not have selected to play.

Algorithm 1 Pseudocode for BTS.

```
Initialize J models, R(τ) ← 0
for t = 1, 2, ..., τ do
    x_a ← feature vector for every arm a
    Randomly select one model j_t out of J
    Play arm with highest reward predicted by j_t
    Observe reward r(t)
    for j = 1, 2, ..., J do
        Update j^th model using x_{a_t} and r(t) w.p. 0.5
    end for
    R(τ) ← R(τ) + r(t)
end for
```

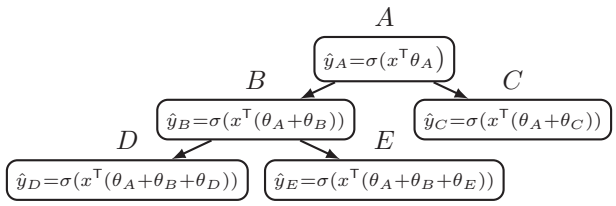

Figure 1: Example Decision Tree with logistic regression models in all nodes. The equations in the nodes denote how a label can be predicted for any instance inside that node. BTS with tree-boosting uses the equations in leaf nodes for predictions.

Tree-Boosting

In Algorithm 1, the J models are shared among all arms; when one model is randomly selected, it is used to make predictions for all arms. It is also possible to have a separate set of J models per arm. The main advantage of this approach is that models may perform better if they can be trained specifically to model the relationship between context vectors and rewards for a single arm, instead of having to generalize to all arms. The main disadvantage of this approach is that a model only gets feedback if its arm was actually played in a given time step, and therefore learns less quickly.

In the setting where arms are leaf nodes of a Decision Tree, the first approach can be viewed as placing a single (shared) model in the root node, and the second approach can be viewed as placing one model in every leaf node. In the literature on offline Decision Tree learning, it is common to place models in every node of a tree, and combine all the models on the path from the root node to a leaf for predictions in that leaf (Quinlan 1992; Malerba et al. 2002; Gama 2004). We propose a novel variant of this idea, suitable for an incrementally updated tree in a real-time setting. The goal is to rapidly learn general models close to the root node which receive updates in a large number of time steps (because they cover a large number of leaf nodes), and slowly learn more specific models close to leaf nodes which can correct errors made by the more general models. This idea is referred to as tree-boosting, because the structure of the tree is exploited for an idea similar to boosting.

For simplicity, suppose that a variant of BTS is used with $J = 1$, and an update probability of 1 per time step. We no longer uses the same model to make predictions for every arm, but make predictions by combining a series of models in nodes on the path from root to leaf. Logistic regression models parameterized by vectors θ are used as base models. They predict class labels from context vectors x using $\hat{y} = \sigma(x^\mathsf{T}\theta)$, where $\sigma(z) = \frac{1}{1+\exp(-z)}$ is the standard logistic function. The rewards observed are not binary, but real-valued. However, because the $amount(x)$ value is known, it is always possible to multiply or divide by the amount to convert between binary class predictions and reward values. Therefore, it is sufficient to train binary class predictors.

An example Decision Tree with tree-boosted logistic regression models is depicted in Figure 1. Every node N has a parameter vector θ_N for the logistic regression model in

that node. Suppose that, at time t, a leaf node L is played. Starting from the root, the context vector x_t and observed reward $r(t)$ are used to update all models on the path to L. These updates are performed using Stochastic Gradient Descent. Consider, for example, that leaf node D in the figure is played. The feedback is used to update θ_A, θ_B and θ_D, in that order. Because θ_A and θ_B are re-used for predictions in node D, this means that the feedback was actually used for three consecutive updates in node D.

6 Semi-Supervised Decision Tree Splits

The FIMT-DD Regression Tree learning algorithm only creates splits in the tree if it considers the number of labeled observations to be sufficiently large such that it can create "optimal" splits with a desired confidence level. If a leaf node L does not observe enough labeled instances (for example, because instances from other leaves keep getting pulled instead), the number of instances in L can grow large. This can result in a wide variety of rewards within the same leaf node, which makes the resulting CMAB problem more difficult. To prevent a single leaf node from representing too many different transactions at the same time, it is proposed to use unlabeled data with an idea based on Semi-Supervised Learning (Zhu 2005) to create extra splits in such nodes.

At certain points in time, the logistic regression models trained as described in Section 5 are used to predict the rewards for *all* transactions in a given leaf node L. This can, for example, be done whenever a new transaction is added to L such that it contains exactly a multiple of 50 instances. These predictions are presented to the FIMT-DD Regression Tree learner, as if they were true labels. This enables FIMT-DD to create new splits in the tree more frequently than if it only based its splits on labels provided by human experts.

7 Experiments

The proposed approach (Tree-Boosted Bootstrap Thompson Sampling on an incrementally generated Regression Tree) is evaluated using a publicly available dataset of real credit card transactions, and data generated by a simulator. The task of the system is to select transactions to investigate from this data at regular points in time. For example, a system can be allowed to select one transaction after every N seconds if the data contains timestamps, or one transaction after every N incoming transactions. The approach is compared to offline models, which are trained to predict the probability of transactions being fraudulent. They multiply this by transaction amounts to compute the predicted rewards, and sort transactions into a priority queue using these predicted rewards. The labeled data used to train these offline models is also used to build a small initial Regression Tree in our approach. The following approaches are evaluated:

- **XGBoost**: An offline model (Chen and Guestrin 2016).
- **Random Forest (RF)**: An offline model (Breiman 2001). RFs were found to outperform other offline models, such SVMs and Neural Networks, in other fraud detection systems (Dal Pozzolo et al. 2014a; Lebichot et al. 2016).
- **Random**: An approach that uses FIMT-DD to construct CMAB arms, but randomly selects arms to play.
- **TBBTS**: Tree-Boosted Bootstrap Thompson Sampling as proposed in this paper, without extra SSL-based splits.
- **TBBTS (SSL)** TBBTS with extra SSL-based splits as described in Section 6.

The main performance criterion is the cumulative reward collected by a system over the duration during which data was collected. An alternative performance criterion is $Precision@K$, which is the ratio of the total number of investigations performed (K) that turned out to be fraudulent.

Real Credit Card Transaction Dataset

The real credit card transaction dataset[1] (Dal Pozzolo et al. 2015b) contains 284,807 transactions, of which 492 were fraudulent. These records were gathered over a period of 48 hours. Every transaction has a timestamp, a monetary amount, and 28 other real-valued, anonymized features.

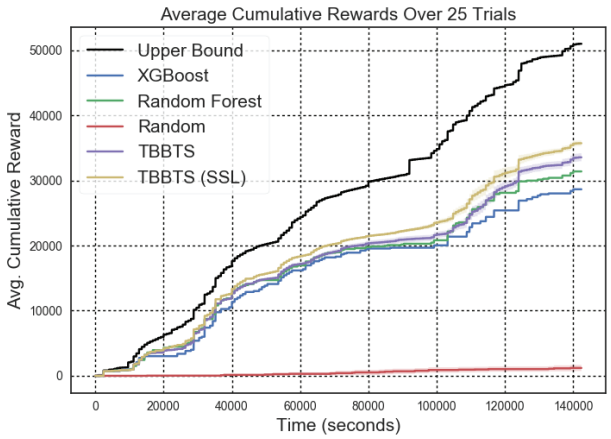

Figure 2: Cumulative Rewards with 800 seconds per investigation, 20K training instances (real data).

Figure 2 depicts 95% confidence intervals for the mean cumulative rewards over time for 25 full runs through this dataset, with an investigation time of 800 seconds per transaction. These 800 seconds result in a realistic total number of investigations according to field experts (Lebichot et al. 2016). The Upper Bound is the performance of a "cheating" system which knows the true class labels of instances, but can still only select one transaction every 800 seconds. At early timestamps the compared approaches all have a similar performance level (except for the Random approach), but TBBTS and TBBTS (SSL) outperform the offline approaches by a statistically significant amount at the later timestamps. This suggests that these approaches are able to improve (relative to the offline models) over time.

MultiMAuS Simulator Data

MultiMAuS (Zintgraf et al. 2017) is a simulator that was built to generate synthetic transaction datasets which are similar to a (not publicly available) real dataset. A major advantage of generating data through such a simulator is that

[1] www.kaggle.com/dalpozz/creditcardfraud

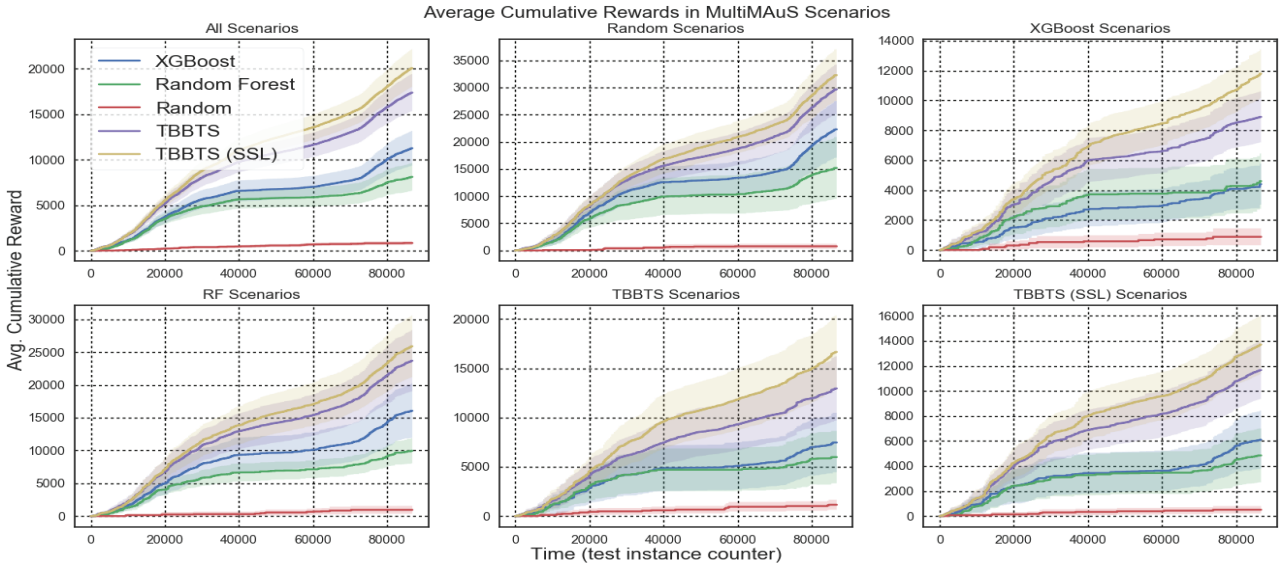

Figure 3: Cumulative Rewards with one investigation every 360 transactions, with approximately 30K training instances. Top-left: all 125 scenarios generated by MultiMAuS. Other plots: 25 scenarios generated against one specific approach.

the data generation process can be modified interactively in response to the behavior of fraud detection models during an experiment. This means that it is possible to simulate concept drift caused by fraudsters changing their behavior when they notice that certain types of transactions get caught.

This type of concept drift was simulated using the following procedure. The natural migration of genuine and fraudulent customers into and out of the system was significantly reduced. Whenever a model catches a fraudulent transaction in an experiment (meaning that a fraudulent transaction was selected for investigation), the associated Card ID is banned in the simulator and a new fraudster is generated. This creates a "survival of the fittest" effect for fraudsters, where new fraudsters are generated whenever old fraudsters get caught, but fraudsters are allowed to remain in the system for a long time if they do not get caught.

This experimental setup makes it unfair to directly compare the performance of different approaches in terms of cumulative rewards to each other, because different approaches can be confronted with significantly different scenarios. For example, the distribution of fraudulent transactions may drift towards primarily low amounts against one approach, and primarily high amounts against another approach. The second approach would then have a significantly larger upper bound on the cumulative rewards it could potentially gather within the same amount of time. Therefore, we stored the datasets generated in 25 trials against every approach as scenarios, and then evaluated the five approaches on all 125 scenarios, treating them as non-interactive datasets.

The top-left plot in Figure 3 depicts 95% confidence intervals for the mean cumulative rewards gathered over all 125 scenarios. Each of the other plots depicts the results obtained by evaluating all approaches on only 25 scenarios generated interactively against one particular approach. For example, the plot titled "Random Scenarios" depicts the results on the 25 scenarios generated against Random. In all plots, the TBBTS and TBBTS (SSL) approaches outperform the offline approaches. The performace of offline models appears to degrade more over time in scenarios generated against stronger approaches than in scenarios generated against weaker approaches, such as Random. The scenarios generated against XGBoost are the only ones where RFs appear to perform slightly better than XGBoost, which indicates that there was indeed pressure on fraudsters in these scenarios to specifically avoid detection by XGBoost.

The $Precision@K$ metrics recorded at the final time steps of all experiments are shown in Table 1. In most cases this metric results in the same ordering of approaches as cumulative rewards, but there are some cases with differences.

8 Conclusions

This paper addresses the problem of deciding which instance to send to a human expert for investigation in a data stream of credit card transactions. Investigating a transaction reveals whether or not it is fraudulent, and results in a reward if it is fraudulent. It can be beneficial to investigate transactions for which a model's predictions are uncertain (exploration), and it can be beneficial to investigate transactions which are predicted with high certainty to be fraudulent (exploitation). We propose to use incremental Regression Trees to cluster transactions according to their predicted rewards. This enables the use of Contextual Multi-Armed Bandit (CMAB) algorithms, with leaf nodes as arms, to address the trade-off between exploration and exploitation. This would be computationally infeasible if every instance were treated as a separate arm. Furthermore, we propose a novel variant of a CMAB algorithm which exploits the structure of the Regression Tree that defines the arms, and use Semi-Supervised Learning to make use of unlabeled

Table 1: 95% conf. intervals for $Precision@K$ at final time step

Experiment	Random	XGBoost	Random Forest	TBBTS	TBBTS (SSL)
Real Data (Figure 2)	0.005 ± 0.002	0.376	0.455	0.669 ± 0.026	0.724 ± 0.017
All Scen. (Figure 3)	0.002 ± 0.001	0.083 ± 0.015	0.054 ± 0.015	0.391 ± 0.040	0.381 ± 0.030
Random Scen. (Figure 3)	0.003 ± 0.001	0.184 ± 0.045	0.120 ± 0.063	0.548 ± 0.075	0.524 ± 0.059
XGBoost Scen. (Figure 3)	0.003 ± 0.001	0.028 ± 0.009	0.025 ± 0.014	0.238 ± 0.065	0.270 ± 0.055
RF Scen. (Figure 3)	0.004 ± 0.001	0.114 ± 0.032	0.049 ± 0.017	0.481 ± 0.103	0.447 ± 0.073
TBBTS Scen. (Figure 3)	0.003 ± 0.001	0.049 ± 0.019	0.037 ± 0.025	0.351 ± 0.093	0.330 ± 0.062
TBBTS (SSL) Scen. (Figure 3)	0.002 ± 0.001	0.042 ± 0.013	0.039 ± 0.024	0.339 ± 0.084	0.333 ± 0.055

data when building the tree. The entire approach can be incrementally updated in real-time, which enables it to adapt more quickly to concept drift, using expert feedback, than models trained offline can.

The approach has been shown to perform competitively in terms of cumulative rewards with commonly used offline approaches, such as Random Forests, on a dataset of 48 hours of real credit card transactions. Additionally, it has been shown to outperform such offline models more convincingly on data generated by a simulator which adds extra concept drift by pressuring fraudsters to adapt their behavior in an attempt to avoid detection by models.

In future work, cost-sensitive variants of the approach could be evaluated in settings where instances have variable investigation costs. It would also be interesting to evaluate different mechanisms for exploration in CMAB algorithms. Combining the approach in an ensemble with offline models may result in a stronger overall system. Finally, the approach could be evaluated in entirely different domains, such as selecting which image to examine next for a medical expert.

Acknowledgements. This work was funded by the INNOVIRIS project C-Cure. Thanks to Luisa Zintgraf for discussions about the MultiMAuS simulator. Thanks to the company (identity undisclosed) that provided the data on which the MultiMAuS simulator is based.

References

Agrawal, S., and Goyal, N. 2013. Thompson Sampling for Contextual Bandits with Linear Payoffs. In Dasgupta, S., and McAllester, D., eds., *Proc. of the 30th Int. Conf. on Mach. Learning*, volume 28 of *Proceedings of Machine Learning Research*, 127–135.

Breiman, L. 2001. Random Forests. *Mach. Learning* 45(1):5–32.

Chen, T., and Guestrin, C. 2016. XGBoost: A Scalable Tree Boosting System. In *Proc. of the 22nd ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, 785–794. ACM.

Dal Pozzolo, A.; Caelen, O.; Le Borgne, Y.; Waterschoot, S.; and Bontempi, G. 2014a. Learned Lessons in Credit Card Fraud Detection from a Practitioner Perspective. *Expert Systems with Applications* 41(10):4915–4928.

Dal Pozzolo, A.; Johnson, R.; Caelen, O.; Waterschoot, S.; Chawla, N. V.; and Bontempi, G. 2014b. Using HDDT to Avoid Instances Propagation in Unbalanced and Evolving Data Streams. In *IJCNN 2014*, 588–594. IEEE.

Dal Pozzolo, A.; Boracchi, G.; Caelen, O.; Alippi, C.; and Bontempi, G. 2015a. Credit Card Fraud Detection and Concept-Drift Adaptation with Delayed Supervised Information. In *IJCNN 2015*, 1700–1707. IEEE.

Dal Pozzolo, A.; Caelen, O.; Johnson, R. A.; and Bontempi, G. 2015b. Calibrating Probability with Undersampling for Unbalanced Classification. In *Symp. on Computational Intell. and Data Mining (CIDM)*, 159–166. IEEE.

Dasgupta, S. 2011. Two Faces of Active Learning. *Theoretical Computer Science* 412(19):1767–1781.

Delamaire, L.; Abdou, H.; and Pointon, J. 2009. Credit Card Fraud and Detection Techniques: a Review. *Banks and Bank Systems* 4(2):57–68.

Eckles, D., and Kaptein, M. 2014. Thompson Sampling with the Online Bootstrap. *ArXiv e-prints*.

Elmachtoub, A. N.; McNellis, R.; Oh, S.; and Petrik, M. 2017. A Practical Method for Solving Contextual Bandit Problems Using Decision Trees. *ArXiv e-prints*.

Gama, J. 2004. Functional Trees. *Mach. Learning* 55(3):219–250.

Garnett, R.; Krishnamurthy, Y.; Wang, D.; and Schneider, J. 2011. Bayesian Optimal Active Search on Graphs. In *Ninth Workshop on Mining and Learning with Graphs*.

Ikonomovska, E. 2012. *Algorithms for Learning Regression Trees and Ensembles on Evolving Data Streams*. Ph.D. Dissertation, Jožef Stefan International Postgraduate School, Ljubljana, Slovenia.

Lebichot, B.; Braun, F.; Caelen, O.; and Saerens, M. 2016. A Graph-Based, Semi-Supervised, Credit Card Fraud Detection System. In Cherifi, H.; Gaito, S.; Quattrociocchi, W.; and Sala, A., eds., *Complex Networks & Their Applicat. V*, volume 693 of *Stud. in Computational Intell.*, 721–733. Springer.

Li, L.; Chu, W.; Langford, J.; and Schapire, R. E. 2010. A Contextual-Bandit Approach to Personalized News Article Recommendation. In *Proc. of the 19th Int. Conf. on World Wide Web*, 661–670. ACM.

Malerba, D.; Appice, A.; Ceci, M.; and Monopoli, M. 2002. Trading-Off Local versus Global Effects of Regression Nodes in Model Trees. In Hacid, M. S.; Raś, Z. W.; Zighed, D. A.; and Kodratoff, Y., eds., *Int. Symp. on Methodologies for Intelligent Syst.*, volume 2366 of *LNCS*, 393–402. Springer, Berlin, Heidelberg.

Quinlan, J. R. 1992. Learning with Continuous Classes. In *Proc. of the 5th Australian Joint Conf. on Artificial Intell.*, 343–348.

Zhou, L. 2015. A Survey on Contextual Multi-armed Bandits. *ArXiv e-prints*.

Zhu, X. 2005. Semi-Supervised Learning Literature Survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison, Madison, Wisconsin, United States.

Zintgraf, L. M.; Lopez-Rojas, E. A.; Roijers, D. M.; and Nowé, A. 2017. MultiMAuS: A Multi-Modal Authentication Simulator for Fraud Detection Research. In *29th European Modeling and Simulation Symp. (EMSS 2017)*, 360–370. Curran Associates, Inc.